Erste Hilfe für den C 64: RENEW

Manchmal läuft auch alles schief! Da waren die Finger wieder einmal schneller als die Gedanken. Man hat NEW eingetippt und erst nach dem Drücken der RETURN-Taste wird einem der angerichtete, ungewollte Schaden klar: Das Programm ist weg! Oder?

Die VC 20-Besitzer können jetzt nur lächeln — wenn sie die Ausgabe 4 kennen. Dort ist nämlich ein Programm abgedruckt, daß ein mit NEW gelöschtes Programm wieder sichtbar macht. Hier ist die angepaßte Version für den C 64.

Eine Programmbeschreibung erübrigt sich, da in der Ausgabe 4 Helmut Welke dies bereits sehr ausführlich gemacht hat. Es sind auch nur sehr kleine Änderungen zu machen. Im Originallisting für den VC 20 (4/84 Seite 89) sind in Zeile 1005 der Wert 207 durch 175 und in Zeile 1015 der Wert 198 durch 166 zu ersetzen und schon läuft's auch im C 64. Bei diesem hilft dieses RENEW auch nach einem Reset oder SYS 64738. Auch dieses Programm kann selbstverständlich mit dem DATA-WANDLER abgespeichert werden.

Listing von »RENEW«

```
100 REM *** RENEW ***
110 FORI=678 TO 755:READA:POKEI,A:NEXT
111 DATA165,43,24,105,4,133,253,165,44,1
05,0,133,254,160,0,177,253,240,8
112 DATA200,192,88,208,247,76,8,175,200,
152,160,0,24,101,253,145,43,133,253
113 DATA144,2,230,254,165,254,200,145,43
,136,177,253,170,200,177,253,240
114 DATA7,133,254,134,253,76,213,2,165,0
,24,105,2,133,45,165,254,32,85,166
115 DATA76,156,166
READY.
```

Datawandler

Mit dem Datawandler ist es auch dem »Nur-Basicprogrammierer« möglich, Maschinensprachprogramme (oder Teile davon), die in Form eines Basic-Laders (also über DATAs und POKE-Befehle) eingetippt wurden, als Maschinensprache abzuspeichern. Dadurch ergibt sich auf der Diskette eine Platz-, beim Laden eine Zeitersparnis.

»Bitte warten — ich lese Daten« — so oder ähnlich wird der »Basicprogrammierer« nach dem Programmstart darauf aufmerksam gemacht, daß bei jedem RUN eines Basicprogramms die Maschinenspracheteile DATA für DATA in die Speicherzellen gePOKEt werden. Warum also nicht die DATAs gleich wieder als Maschinensprache abspeichern! Aber es ergibt sich nicht nur eine Zeitersparnis beim Programmlauf, sondern auch beim Laden, da auf der Diskette oder Kassette weniger Speicherplatz benötigt wird (zum Vergleich: der Schatzsucher aus der Ausgabe 6 belegt auf der Diskette als Basicprogramm 72 Blöcke, als Kombination Basic/Maschinensprache nur noch 44 — außerdem startet dieses Programm dann in Sekundenbruchteilen).

Darüber hinaus bestehen viele Hilfsprogramme, die auf Maschinenroutinen zurückzugreifen, nur aus der Zeile »FORI=xTOxx:READx:POKEI,x:NEXTI« und vielen, vielen DATAs. Derartige Programme bieten sich für den Datawandler von selbst an, da sie nach der Umformung geladen werden können, ohne ein eventuell im Basicspeicher stehendes Programm zu zerstören.

Zum Programm selbst

Vor dem Start des Datawandlers müssen die Daten im Speicher stehen — falls sie nur in Form von DATA-Zeilen vorhanden sind, wird in Zeile 60130 eingefügt:

```
60130 FORI=AA TO EA:READX:POKEI,X:NEXTI
```

In den Zeilen 60030 — 60040 wird (in dezimaler Form) die Anfangs- und Endadresse abgefragt, unter der die DATAs »abgelegt« sind und den Variablen AA beziehungsweise EA zugeordnet. Die Zeilen 60050 bis 60100 dienen der Abklärung, ob das abzuspeichernde Programm auch wieder an die Adresse geladen werden soll, wo derzeit die DATA stehen (durch das Verschieben der Ladeadresse ist auch ein leichteres Experimentieren in den Autostartbereichen möglich, die dem Programmierer gelegentlich nach den ersten »POKEs« das Konzept aus der Hand nehmen).

Wird die Abfrage mit »N« beantwortet, so wird nach dieser späteren Ladeabdresse gefragt — lautet die Antwort »J«, so wird die Ladeadresse = derzeitige Anfangsadresse (LA=AA) und nach dem Namen gefragt, unter dem das Programm nun

abgespeichert werden soll (Zeile 60120).
Zeile 60130 — siehe oben.

So dann wird der Floppykanal geöffnet (Zeile 60140), die dezimalen Eingaben auf »Diskettenformat« gebracht (Zeile 60150), damit die ersten beiden Bytes auf der Diskette als Ladeadresse geschrieben werden können (Zeile 60160). In den Zeilen 60170 bis 60190 werden schließlich die Daten ausgelesen und direkt auf die Diskette geschrieben. Die übrigen Zeilen dienen der Abfrage des Fehlerkanals der Floppy beziehungsweise schließlich der »Fertig«-Meldung.

Weitere Hinweise:
1) Das Programm ist in dieser Form sowohl für den VC 20 als auch für den C 64 verwendbar.
2) Bei entsprechender Abänderung des OPEN-Befehls sollte auch die Abspeicherung auf Kassette möglich sein (die Zeilen 60200 bis 60240 entfallen dann).
3) Vor dem Abspeichern sollten die DATAs natürlich korrekt sein, da nach dem Abspeichern eine Überprüfung noch schwieriger ist.
Bei Basicprogrammen sollten deshalb der Basicteil und die DATAs zunächst unabhängig voneinander eingegeben, zum Probelauf mit »MERGE« zusammengefügt und bei Fehlerlosigkeit der DATA-Teil dann entsprechend abgespeichert werden.
4) Die »eigenständigen« Maschinenprogamme werden dann mit LOAD'xy',8,1 geladen und mit dem SYS-Befehl gestartet.
Bei Basicprogrammen sollte dann (sofern nicht ein Autostartprogramm zum Laden aller Teile verwendet wird), die erste Programmzeile lauten:
IFA=0THENA=1: LOAD »Name des Maschinenspracheteils«,8,1
— das klingt zwar paradox, aber es funktioniert: nach dem RUN wird dann geladen und gestartet.
Daß in einem solchen Basicprogramm alle READ-Befehle etc. ausgebaut werden müssen, versteht sich wohl von selbst.

(Uwe Christian Parpart/gk)

Listing »Datawandler«

```
60000 REM *** DATAWANDLER ***
60010 REM *** (C) UWE CHR. PARPART ***
60020 PRINT"■■    *** D A T A W A N D L
E R ***"
60030 INPUT"■ JETZIGE ANFANGSADRESSE";AA
60040 INPUT"■     JETZIGE ENDADRESSE";EA
60050 PRINT"■ IST DIE ANFANGSADRESSE IDE
NTISCH"
60060 PRINT"■ MIT SPAETERER LADEADRESSE
(J/N)?"
60070 GETA$:IFA$=""THEN60070
60080 IFA$="J"THENLA=AA:GOTO60120
60090 IFA$="N"THEN60110
60100 GOTO 60070
60110 INPUT"■   SPAETERE LADEADRESSE";LA
60120 INPUT"■ NAME DES PROGRAMMS";L$
60130 OPEN1,8,1,L$+",P,W"
60140 HB=INT(LA/256):LB=LA-HB*256
60150 PRINT#1,CHR$(LB);CHR$(HB);
60160 FORI=AATOEA
60170 PRINT#1,CHR$(PEEK(I));
60180 NEXTI:CLOSE1
60190 REM *** ABFRAGE FEHLERKANAL ***
60200 OPEN1,8,15
60210 INPUT#1,A,B$,C,D
60220 PRINTA;B$;C;D
60230 CLOSE1
60240 PRINT"■■■■F E R T I G !■"
60250 END
READY.
```

Simons Basic: Befehle, die nicht im Handbuch stehen

Als Ergänzung zu den Artikeln über Simons Basic in den Ausgaben 4/84 und 5/84 wollen wir in dieser Ausgabe noch einige Befehle und Besonderheiten aufführen, die nicht in jedem Handbuch stehen.

Zunächst die zusätzlichen Befehle in alphabetischer Reihenfolge:

BCKGNDS
Syntax: BCKGNDS f1,f2,f3,f4
— f1: normale Hintergrundfarbe
— f2: Hintergrundfarbe der Zeichen mit SHIFT-Taste
— f3: Hintergrundfarbe der REVERS-Zeichen und des Cursors (nicht der Schriftfarbe)
— f4: Hintergrundfarbe für Zeichen mit SHIFT-Taste im REVERS-Mode
Semantik: BCKGNDS legt die Hintergrundfarben fest und schaltet auf ECM (Extended-Color-Mode), dabei werden von jedem Zeichen zwei Bit vom ASCII-Code abgezweigt: es steht somit nicht mehr der gesamte Zeichensatz zur Verfügung.
NRM macht BCKGNDS rückgängig

COLOUR
Syntax: COLOUR, rf, hf
— rf: Rahmenfarbe
— hf: Hintergrundfarbe
Semantik: COLOUR setzt Rahmen- und Hintergrundfarbe und erspart somit das lästige POKE 53280,rf : POKE 53281,hf.

DISABLE
Syntax: DISABLE
Semantik: Setzt ON KEY-Anweisung außer Kraft

GRAPHICS:
Syntax: GRAPHICS
Semantik: Liefert Konstante $D000 = 53248; Adresse VIC

NRM
Syntax: NRM
Semantik: NRM macht MEM und BCKGNDS rückgängig.

ON KEY
Syntax: ON KEY Stringausdruck, diverse Anweisungen
Semantik: Wird eine Taste gedrückt, die im Stringausdruck des ON KEY-Befehls enthalten ist, so wird in den Anweisungsteil verzweigt. Die Tastatur wird dabei vor jedem Befehl abgefragt. Ein unbedingter Sprung erfolgt, wenn im Stringausdruck eine »eckige Klammer zu« ($5D) enthalten ist.

RESUME
Syntax: RESUME
Semantik: RESUME funktioniert nur nach ON KEY. Bei RESUME wird das Programm beim ursprünglichen Befehl fortgesetzt. RESUME entspricht somit dem RETURN bei GOSUB.